Uluslararası İleri Doğa Bilimleri ve Mühendislik Araştırmaları Dergisi
Sayı 7, S. 5-10, 2, 2023
© Telif hakkı IJANSER'e aittir
Araştırma Makalesi

IJANSER
https://as-proceeding.com/index.php/ijanser
ISSN: 2980-0811

International Journal of Advanced Natural Sciences and Engineering Researches
Volume 7, pp. 5-10, 2, 2023
Copyright © 2023 IJANSER
Research Article

Achieving Loose Coupling and High Quality through Migrating the Monolithic Applications to Microservices Applications

Ayisha Al-Saidi [1], Zuhoor Al-Khanjari [1]

[1]*Department of Computer Science, College of Science, Sultan Qaboos University, Oman, Muscat*

[1]s128554@student.squ.edu.om *Email of the corresponding author*

ATIF/REFERENCE: Al-Saidi, A. & Al-Khanjari, Z. (2023). Achieving Loose Coupling and High Quality through Migrating the Monolithic Applications to Microservices Applications. *International Journal of Advanced Natural Sciences and Engineering Researches*, 7(2), 5-10.

Abstract **–** The growing demands for available data in today's applications from small to complex systems have increased every day. It has forced large enterprises to look for such modern technologies that can provide them with more value, benefits, enhancement, and management of the systems. For illustration, it imposed them to use more manageable architectural design besides a powerful software development process. That is to meet the needs of continuous development, maintenance, and deployment of the current applications in order to achieve the main requirements of the end users like high quality, availability, scalability, performance, and security of the applications. However, those are the main reasons to adopt Microservices and DevOps in current systems and migrate the Monolithic-based Architecture that has faced problems with those requirements as it grows over the years into Microservices. This paper aims to propose a fundamental approach to transfer the existing Monolithic-based Applications to Microservices-based Applications to achieve high quality and the loose coupling between services for current Applications.

Keywords – Monolithic Architecture, Microservices Architecture (MSA), DevOps, Migration, Quality.

I. INTRODUCTION

In traditional applications which are called Monolithic Applications, the fundamental components of the system are developed, packaged, and deployed as a single unit in the same program using the same language and framework [1], [2]. To give a clear understanding, in Java applications, the whole system is packaged as WAR files or JARs and deployed in the same application using a server called Tomcat [1]. Thus, the application is easy to develop and test during the early stages. However, as it grows bigger and bigger over years, it becomes more complicated. For end-to-end testing, the application can be launched on the browser and can be validated using Robot-framework (ex. Selenium lib) [1]. The small applications can be scaled easily using a load balancer by having multiple runnable copies. But for complex systems as scalability, flexibility, maintenance, and time to value are very crucial factors, and it is very hard to maintain and get control. In other words, large applications are difficult to scale up because all components in the applications are closely coupled and connected as well as have overlapping responsibilities. Furthermore, since the structure of the Monolithic applications is highly coupled, and components depend on each other, when any failure happened on one module or component, the overall system needs to be updated, maintained, and deployed. Thus, the main bottlenecks and challenges of Monolithic applications can be pointed out as scalability, slow development, and difficulty in maintenance and deployment. In general, all the aspects of making changes or developing in

Monolithic applications get complicated and impediment over years.

The new architecture of Microservices can be used to overcome these limitations and bottlenecks of the Monolithic. This can be done by decomposing the system into a set of small and independent services that are called Microservices that can communicate through passing messages using RESTful APIs. So, Microservices Architecture is an approach to software architectural design which is responsible to build and deploying distributed applications as small, independent, and deployable services where each service can run as a unique process and communicate with others using lightweight mechanisms [1]. In addition, each service can be programmed using different programming languages and stored through different data storage technologies [3]. There are several companies, such as Netflix, Amazon, and The Guardian that have successfully utilized Microservices in their large-scale software systems [3]. Moreover, some other companies are currently refactoring of their back-end systems to Microservices-based systems to improve maintainability and scalability [4].

The rest of this paper is structured as follows: Parts II, III, and IV discuss the main concepts of the paper which are Monolithic Application, Microservices Architecture, and DevOps. Part V analyses and discusses the related works. Part VI provides the mechanism which is used to transfer Monolithic applications into Microservices applications. The last part is the conclusion and future work of this paper.

II. Monolithic Architecture

Monolithic-based architecture is the common and simplest way to develop and deploy different web applications as a single unit that contains all the responsibilities and shares the same database [1], [2], [5], [6] as shown in Fig 1. Also, those applications have a big codebase because it developed as a single package [2], [5].Therefore, the applications may become too complex as well as difficult to understand. However, this can slow down the development of applications and avoid making updates. As a result, Monolithic Applications face many challenges to make changes, scale up, fix errors, and recover from failures. Moreover, to deploy one update of the system, the entire application must be deployed as well, and this can create some difficulties to track the impacts of the update for the whole application [5]. In other words, scaling up of Monolithic Applications is frequently very complicated and tough because making a single update for any module or one part of the system requires scaling and updating the entire application as well as deploying the whole system [7]. Furthermore, as the Monolithic Applications grow up and their size becomes too large during the many years of development and scaling, it becomes very hard to maintain and make changes to them, so it is the right time to migrate to a new architectural style that has called Microservices in order to overcome these challenges and get the great benefit and values of using Microservices [7]. Large companies such as Amazon, Netflix, LinkedIn, and SoundCloud have made the migration to Microservice Architecture because their existing Monolithic Applications were too tough to scale, maintain, develop, and deploys [7].

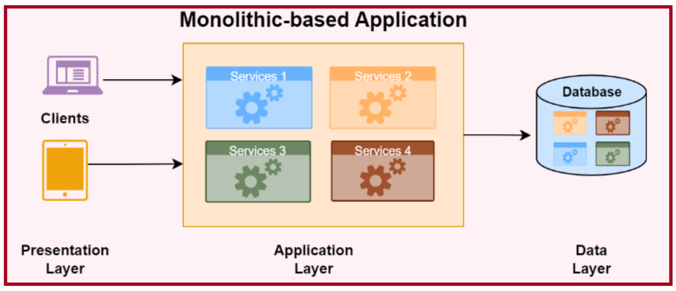

Fig. 1 Monolithic Architecture

III. Microservices Architecture (MSA)

Microservices which were first described by Lewis and Fowler [8] are referred to as a set of small and autonomous services. Each service runs as a unique process and has a single responsibility inside the system to serve a specific task and goal. Moreover, each service can be developed, maintained, and deployed separately without affecting other services. Accordingly, Microservices Architecture in expected to resolve the main downsides of Monolithic applications which are related to scalability, flexibility, maintenance, and continuous delivery of the system [2]. However, Microservices have high cohesion and loose coupling, thus it is robust to make changes and easy to update and maintain , [1], [2], [9]. Since the services are independent and small, they can be scaled up easily as well as achieve continuous delivery and deployment as required [1], [2], [9]. In addition, the developers of Microservices can use different technological stacks in the development process [1], [10]. They can use different languages, data stores, and technologies for each microservice [1], [10]. However, there are some drawbacks to Microservices Architecture. For illustration, Microservices required to use more resources and tools to achieve architectural flexibility which led to increasing the complexity of the system [10], [11] . As Monolithic Systems become large and more complex to deal with, some organizations take the decision to break their systems by migrating from Monolithic applications into Microservices applications [5] for many critical reasons. The main reasons to obtain Microservices are high availability, maintenance, flexibility, scalability, easier infrastructure management, compliance with the latest security standards, and combined flow of

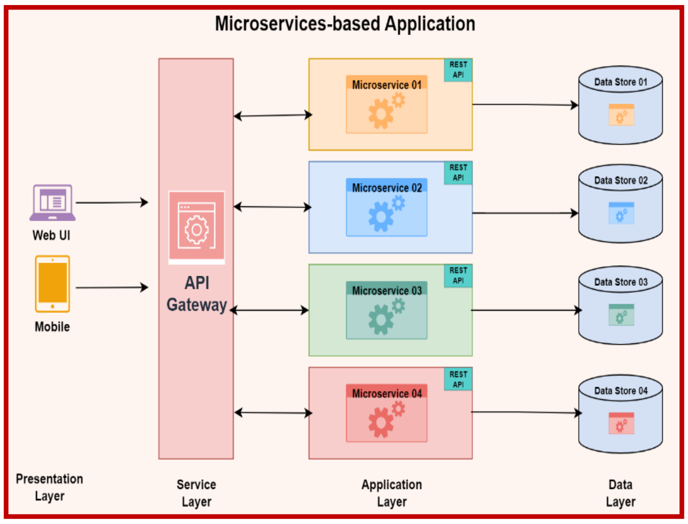

Fig. 2 Microservices Architecture (MSA)

development and operation, which is called DevOps [9], [10]. However, Fig. 2 clearly shows the Microservices Architecture of the application.

IV. DevOps

Since the major goals of Microservices are increasing productivity, availability, scalability, flexibility, rapid deployment, and delivery, Microservices emerged with DevOps practices to create a rapid and automated deployment of software [10]. However, DevOps is a combination of some practices and tools that are used to facilitate the process of development, integration, testing, and deployment of high-quality software [12], [13]. It aims to break down the boundaries between the Development and Operations of the system to increase the ability to deploy it faster with high quality. Combining DevOps method and Microservices can add more benefits and high levels of efficiency to the application [12]. For example, building Microservices through DevOps practices and tools can provide automation techniques for the application like continuous delivery and deployment using pipelines. In addition, they can provide for Microservices continuous integration, adding new features in a safe and secure way, and also can recover quickly from failures [12]. To give a clear example, DevOps can provide automated security practices for Microservices like dependency scanning and static analysis using some valuable tools like Selenium, Appium, and Cucumber [14], [15]. Those tools help team members to automate most test cases during the testing stage.

V. Related Works

According to IDC (International Data Corporation), in 2022, about 90% of the newest applications were expected to be established on Microservices Architectures [16]. However, the emergence of Microservices and DevOps can increase the agility and flexibility of the applications and enable organizations to rapidly bring their services and products to the market [16]. It is known that building a new application from scratch based on Microservices Architecture is very expensive and time-consuming work to go through. Instead, the enterprises can reuse and migrate the existing system into Microservices following specific approaches or processes. In fact, many enterprises have been taking the decision to migrate their existing applications to Microservices Architectures (MSA). However, adopting such applications is not a simple task to do since it requires refactoring the existing Monolithic Application to migrate the application successfully [5]. Before taking this decision and starting this journey of migration, it needs to answer many questions as well as understand the application accurately. However, there is some research that identifies the challenges of refactoring or migrating from Monolithic Applications to Microservices as in [7]. This paper makes a step forward by first discussing the reasons that force companies to migrate from Monolithic to Microservice and then describes the different challenges that the enterprises may face during the migration and how to be solved or avoided before or during using Microservices. Some challenges are related to:

1. How to handle failures of one or more service.
2. The communication between services.
3. Performance issues.
4. How to handle the orchestration of the Microservices in the system.
5. Which tools to use in this situation.

In addition, the most important challenge that needs to be considered and studied well is how to refactor the Monolithic Applications into microservices, processes, and tools without affecting the current system. Also, it needs to define the process of splitting the existing services into Microservices.

Other studies have proposed some methods, processes or approaches to be followed for the process of migrating Monolithic applications into Microservices like what included in [9]. This paper has compared and classified some refactoring approaches proposed in the academic literature. For illustration, Escobar, et. al. in [17] have defined a Microservice diagram to suggest some alternatives on how to split the existing applications into small pieces of code and services called Microservices. Moreover, in the paper of [18], Ahmadvand, et. al. has defined a conceptual methodology to break down the Monolithic Apps into Microservices that mainly aim for reconciled security and scalability trade-offs. There are some other studies that have discussed transferring, migrating, decomposing, or refactoring Monolithic to Microservices like [1], [16], [19]–[22].

In general, this topic is still not mature yet, and the migration of Monolithic to Microservices besides the architectural refactoring is still a trend and challenging topic those days. Thus, there is a crucial need for more studies to conduct on this topic to find applicable solutions to those challenges because many companies are facing some obstacles to stepping forward [9]. This paper attempts to support some sides of this research gap which is trying to find a fundamental approach to transferring Monolithic-based Apps to Microservices-based Apps. This is to achieve high quality and loose coupling between services with the help of DevOps practices and tools.

I. TRANSFORM MONOLITHIC APPS INTO MICROSERVICES

Mainly, the Monolithic can be simple to develop, deploy, test, maintain, and scale applications while the size of the codebase is still small and modest. However, the Monolithic applications that have grown over years may become more complex and complicated to deal with. Therefore, making changes, scaling up, and deploying such applications can be extremely painful [9]. Consequently, to overcome this gap in the existing monolithic applications, the various organizations take the decision to break down their systems into small and independent services called Microservices. This new strategy creates other critical challenges that need to be well addressed to build successful applications based on Microservices Architecture. The most important challenge is how to split the Monolithic into Microservices. In other words, it is how to transfer the existing systems into Microservices based Architecture without the need to build those Microservices from the scratch. This paper proposes a fundamental approach that can guide the enterprises on their first steps towards migrating the existing Monolithic Applications into Microservices incrementally using DevOps method as clearly shown in Fig.3.

Mainly, the migration process and refactoring of the Monolithic Applications involves upgrading of the architectural design, updating the code, handling the data, migrating deployment environment, and changing the interaction modes [20]. These steps can be considered as the essential steppingstone toward transforming any Monolithic Application into Microservices in an appropriate way. This means that the services to be transferred into Microservices with their connections should be clearly defined and analysed before and after the migration can eliminate the negative effects that may occur during migration. To give a clear illustration, the hidden dependencies among the modules and services or tight coupling between services may negatively affect the designed system.

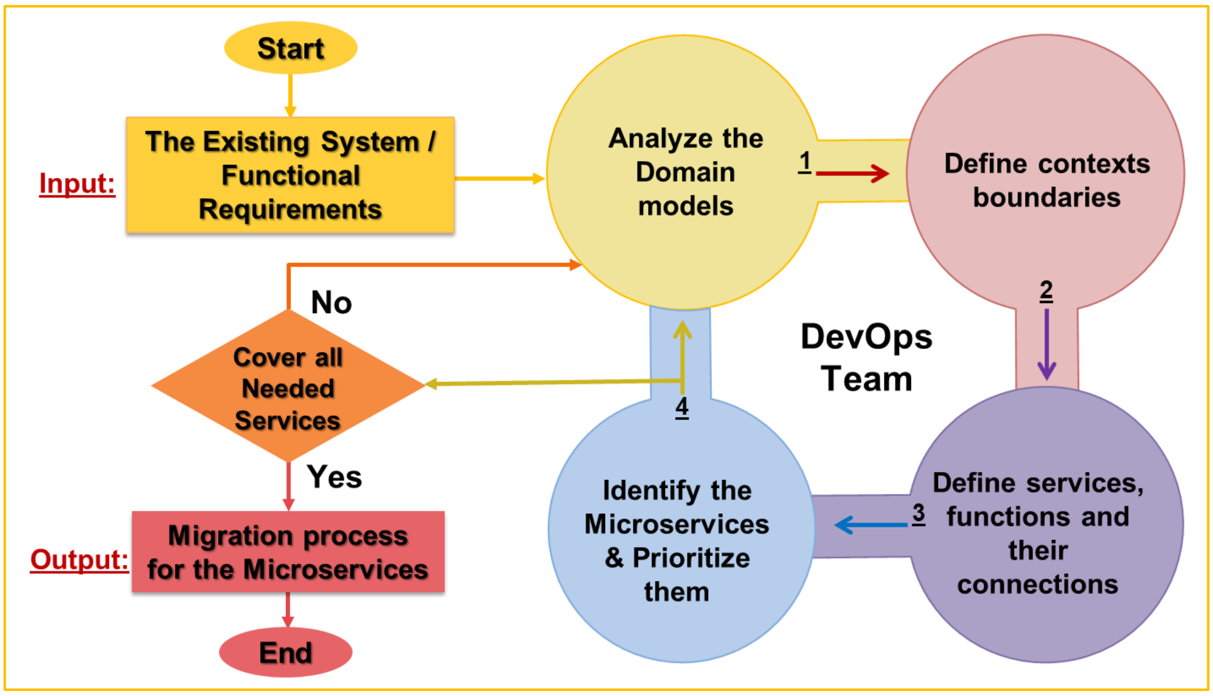
Start
Input:
The Existing System / Functional Requirements
Analyze the Domain models
1
Define contexts boundaries
2
No
Cover all Needed Services
Yes
4
DevOps Team
Output:
Migration process for the Microservices
End
Identify the Microservices & Prioritize them
3
Define services, functions and their connections

Fig. 3 The Fundamental Process of Migrating the Monolithic-based Applications into Microservices-based Applications.

Overall, the output of these fundamental steps can be considered as a migration process for the Microservices to be followed in the process of refactoring the existing Monolithic Applications into Microservices-based Applications using MSA. Besides that, there is another applicable and effective migration process for the shared database to achieve consistency and high-quality transformation for data which was proposed by Volynsky et al. in [2]. This paper has defined a robust framework to migrate the shared database of the Monolithic Applications into a database-per-microservice pattern besides using the Saga pattern for Microservices-based Architecture. These fundamental steps can provide a successful plan to start and migrate Monolithic Apps to Microservices Apps.

VII. Conclusions

Generally, the process of migrating Monolithic-based applications into Microservices-based Architecture is considered a form of application modernization. It means that enterprises should move to Microservices by incrementally refactoring the Monolithic Applications into a set of small services called Microservices following an accurate process or approach instead of rewriting the application from the scratch and wasting time and effort. This paper concludes that this migration is not an easy and obvious task to go through. Therefore, the organizations that make this critical decision need to be well-aware and clearly understand what and how to do the migration in an appropriate way before starting the journey in the real world to avoid any possible failures during the migration process. Also, it may require a lot of time and effort as well as a good experience to overcome the challenges and achieve the goal successfully. This paper analysed and proposed an effective approach of how to migrate Monolithic-based Applications into Microservices-based Applications using a powerful fundamental process to be followed.

Acknowledgment

The authors would like to thank those who supported them during the period of conducting this study, particularly the academic and technical support of the Department of Computer Science, College of Science at Sultan Qaboos University. Many thanks to SQU for the scholarship that giving to support the first author in conducting this study.

References

[1] Sarita and S. Sebastian, “Transform Monolith into Microservices using Docker,” in *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, Aug. 2017, pp. 1–5. doi: 10.1109/ICCUBEA.2017.8463820.

[2] E. Volynsky, M. Mehmed, and S. Krusche, “Architect: A Framework for the Migration to Microservices,” in *2022 International Conference on Computing, Electronics & Communications Engineering (iCCECE)*, Aug. 2022, pp. 71–76. doi: 10.1109/iCCECE55162.2022.9875096.

[3] F. Tapia, M. Á. Mora, W. Fuertes, J. E. Lascano, and T. Toulkeridis, “A Container Orchestration Development that Optimizes the Etherpad Collaborative Editing Tool through a Novel Management System,” *Electronics*, vol. 9, no. 5, p. 828, May 2020, doi: 10.3390/electronics9050828.

[4] A. Bucchiarone, N. Dragoni, S. Dustdar, S. Larsen, and M. Mazzara, “From Monolithic to Microservices: An Experience Report from the Banking Domain,” *IEEE Software*, vol. 35, pp. 50–55, May 2018, doi: 10.1109/MS.2018.2141026.

[5] L. Matlekovic and P. Schneider-Kamp, “From Monolith to Microservices: Software Architecture for Autonomous UAV Infrastructure Inspection,” in *Embedded Systems and Applications*, Mar. 2022, pp. 253–272. doi: 10.5121/csit.2022.120622.

[6] P. Di Francesco, P. Lago, and I. Malavolta, “Migrating Towards Microservice Architectures: An Industrial Survey,” in *2018 IEEE International Conference on Software Architecture (ICSA)*, Apr. 2018, pp. 29–2909. doi: 10.1109/ICSA.2018.00012.

[7] M. Kalske, N. Mäkitalo, and T. Mikkonen, “Challenges When Moving from Monolith to Microservice Architecture,” 2018, pp. 32–47. doi: 10.1007/978-3-319-74433-9_3.

[8] J. Lewis and M. Fowler, “Microservices,” *martinfowler.com*, Mar. 25, 2014. https://martinfowler.com/articles/microservices.html (accessed Nov. 02, 2022).

[9] J. Fritzsch, J. Bogner, A. Zimmermann, and S. Wagner, “From Monolith to Microservices: A Classification of Refactoring Approaches,” vol. 11350, 2019, pp. 128–141. doi: 10.1007/978-3-030-06019-0_10.

[10] C.-Y. Fan and S.-P. Ma, “Migrating Monolithic Mobile Application to Microservice Architecture: An Experiment Report,” in *2017 IEEE International Conference on AI & Mobile Services (AIMS)*, Jun. 2017, pp. 109–112. doi: 10.1109/AIMS.2017.23.

[11] F. Auer, V. Lenarduzzi, M. Felderer, and D. Taibi, “From monolithic systems to Microservices: An assessment framework | Elsevier Enhanced Reader,” vol. 137, no. 106600, Apr. 2021, doi: https://doi.org/10.1016/j.infsof.2021.106600.

[12] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, “DevOps,” *IEEE Software*, vol. 33, no. 3, pp. 94–100, May 2016, doi: 10.1109/MS.2016.68.

[13] “Microservices and DevOps: Better together,” *MuleSoft*. https://www.mulesoft.com/resources/api/microservices-devops-better-together (accessed Oct. 26, 2022).

[14] A. R. Varma, “Major DevOps Practices To Consider While Implementing Microservices,” *DEVOPS DONE RIGHT*, Jun. 02, 2021. https://blog.opstree.com/2021/06/02/major-devops-practices-to-consider-while-implementing-microservices/ (accessed Feb. 21, 2023).

[15] S. Wickramasinghe, “The Role of Microservices in DevOps,” *BMC Blogs*, Aug. 12, 2021. https://www.bmc.com/blogs/devops-microservices/ (accessed Oct. 20, 2022).

[16] S. Newman, *Monolith to microservices: evolutionary patterns to transform your monolith*. O’Reilly Media, 2019.

[17] D. Escobar *et al.*, “Towards the understanding and evolution of monolithic applications as microservices,” in *2016 XLII Latin American Computing Conference (CLEI)*, Oct. 2016, pp. 1–11. doi: 10.1109/CLEI.2016.7833410.

[18] M. Ahmadvand and A. Ibrahim, “Requirements Reconciliation for Scalable and Secure Microservice (De)composition,” in *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*, Sep. 2016, pp. 68–73. doi: 10.1109/REW.2016.026.

[19] G. Mazlami, J. Cito, and P. Leitner, “Extraction of Microservices from Monolithic Software Architectures,” in *2017 IEEE International Conference on Web Services (ICWS)*, Jun. 2017, pp. 524–531. doi: 10.1109/ICWS.2017.61.

[20] R. Belafia, P. Jeanjean, O. Barais, G. Le Guernic, and B. Combemale, *From Monolithic to Microservice Architecture: The Case of Extensible and Domain-Specific IDEs*. 2021, p. 463. doi: 10.1109/MODELS-C53483.2021.00070.

[21] M. Barbosa and P. Maia, *Towards Identifying Microservice Candidates from Business Rules Implemented in Stored Procedures*. 2020. doi: 10.1109/ICSA-C50368.2020.00015.

[22] Z. Ren *et al.*, “Migrating Web Applications from Monolithic Structure to Microservices Architecture,” in *Proceedings of the Tenth Asia-Pacific Symposium on Internetware*, Beijing China, Sep. 2018, pp. 1–10. doi: 10.1145/3275219.3275230.